APL Library

Bill Daly

This manual documents the APL-Library, a collection of useful functions for the APL programmer.

Published by Daly Web and Edit, Inc.

Table of Contents

APL Library

A collection of usefull functions for the APL programmer.

1 Assert, Testing source Code

This library contains functions to perform unit testing. There are five basic functions, asserttoScreen, assertreturn, assertniltoScreen, assertnilreturn and asserterr. These functions will execute their right argument (the test) and compare its results to the left argument.

There are two functions for the environment, assertsetup and assertcleanup.

- Function assertsetup commands. Execute a list of commands to setup for testing
- Function assertcleanUp commands. Execute a list of commands to clean up after testing
- Function result asserttoScreen test. Prints a message to the screen indicating whether the test succeeded or failed.
- Function b←result assertreturn test. returns a Boolean value indicating whether the test succeeded or failed.
- Function b ← eval_function assertniltoScreen test. Function to test a function without a return value. Such a function must have some side effects as it has no actual effect. One must write a function to test for the side effects and return True or False. Supply the name of that function as a character string.
- Function b ← eval_function assertnilreturn test. Like assertniltoScreen, returns true or false rather than cluttering up your screen.
- Function assertmessage message. A simple (minded) function to display a message
- Function error asserterror test. A Function to test that an error has occurred. Right now this is empty function. When I get a bright idea on how to do it I will.
- Function b←result assert01 test. A helper function to recursively evaluate nested test results. Navigates through the complexity of nested array to define what equals actually means.

2 cfg_file — Windows style configuration files

cfg_file parses Windows style configuration files (ini files). These are text files, usually with a suffix of ini, used by various programs to store configuration information.

Ini files are broken down into sections of name–value pairs. This workspaces stores this information in a lexicon of lexicons (see workspaces lex, lex1 and cl). That is each section is separate item in the first level lexicon and each name–value pair in the section is a separate item in the second level of the lexicon.

`lex` ← *cfgparse_file name* [Function]

Reads an ini file and return a two level lexicon. An optional left argument supplies the character used to begin comment lines in the file.

3 Date, an implementation of Lillian dating

This workspace implements Lillian dating, that is storing dates as the number of days from 10/15/1582. It was proposed by IBM in 1986, and named after Aloysius Lilius who devised the Gregorian Calendar.

Lillian dating simplifies date arithmetic as any date is stored as a simple integer.

This workspace contains the following functions:

- Function int ← datelillian Date

  Returns a Lillian date for a three element vector of year,month,day.
- Function vector ← dateunlillian Lillian_date

  Returns a vector of year,month,day from a Lilian date.
- Function vector ← locale dateparse date_string

  Returns a vector of integers for year, month and day. If dateparse is unable to parse the string it will return an error message.

The locale is a lexicon of the following key-value pairs. It must be defined for your location. We've defined one dateUS.

- months (The months spelled out and in order)
- MTH (The months abbreviated and in order)
- weekdays (The days of the week spelled out and in order)
- wkd (The days of the week abbreviated and in order)
- days (The days of the month)
- two-digit-cutoff (The years less than this are in the last century).
- leap-month (An integer for the month which has the leap-day)
- month_pos (The position of the month in the numeric dates eg. 4/5/2016)
- year_pos (The position of the year in numeric dates)
- day_pos (The position of the year in numeric dates)
- epoch (1582 10 15 – unless one wants something other than lillian dates)

The follwing variables are globally defined:

- Variable dateUS A locale lexicon for US usage
- Variable datecal An 2 12 shaped array where line one is the days of each month in a leap year and line two the days of each month in a normal year.
- Variable datedates A lexicon of two name-value pairs. 'Year 0' is 1200. That is the previous year divisible by 400. (The essence of the Gregorian calendar reform).

  'Pre lillian' is the number of days from 1199 12 31 to 1582 10 15.

  Its best not to ask why this is needed.

4 Document Object Model for APL

This workspace provides an incomplete implementation of w3.com's Document Object Model (DOM). The DOM creates and manipulates a graph database from an xml file. The specification leans heavily on object oriented programing constructs.

This implementation provides a functional programming model with a function naming scheme to identify the objects in the specification. So that functions in the domnode family are methods specified for the node object and domdocument functions are methods specified for the document object. Creation methods require a left argument of the name of the document variable, an idea not fully implemented in this version.

The graph database design departs from the DOM in that children of a node are stored in an apl vector and the node methods firstChild, lastChild and nextSibling have not yet been implemented.

Traversal of the graph is best illustrated by the function domnodetoxml. That function recursively traverses the graph returning the variable xml, viz.:

```
xml←xml,domnodetoxml ¨ domnodechildren node
```

To build a DOM use domparse:

```
)copy 5 FILE_IO
dv ← 'dv' domparse FIOread_file 'ADom.xml'
```

To look at a DOM graph use domnodetoxml:

```
←domnodetoxml node
```

5 finance – Cash flow and present value

The finance workspace provides functions useful in understanding the cash flow and cash requirments of an enterprise and for planning and managing that cash flow.

5.1 Simple amounts

These functions work on a single cash payment. Each function expects a right argument vector of amount, interest rate, and number of periods.

`future_amt` ← *fincompoundValue arg* [Function]

Calculate the future value of a single sum. All arguments are made in a single right-argument vector of cash invested, interest rate per period, and number of periods.

Interest rates are generally quoted at an annual rate ignoring the effects of compounding. Therefore $100 invested at 12% per annum and compounded monthly would yeild $112.68 from entering

```
      fincompoundValue 100, (.12÷12) 12
112.68
```

`amt` ← *finpresentValue arg* [Function]

Function calculates the present value of a single sum payable in n periods. The right argument is assembled as with fincompoundValue. This is the reverse of fincompound value:

```
      finpresentValue 112.68 .01 12
100
```

5.2 Periodic payments

These functions work on a flow of cash. For instance a mortgage (called an anuity here) is usually a loan of a specific sum (negative cash flow) followed by monthly payments and a fixed amount (positve cash flow). For these function the following datum appear

`pay` [Variable]

The periodic payment

`i` [Variable]

The interest per period

`n` [Variable]

The number of periods.

`amt` ← *finpresentValueAnnuity vector of pay i n* [Function]

Function caluclates the present value of an annuity, that is the amount of a loan today in exchange for a payment in each of N periods. The right argument is a vector of the payment, interest rate, and number of periods, viz.:

```
      finpresentValueAnnuity 100 .01 360
9721.83
```

`amt` ← *fincompoundAnnuity vector of pay i n* [Function]

Function calculates the future value of an annuity. That is the amount in a savings account after n periods of depositing the same amount.

```
      fincompoundAnnuity 100 .01 360
352991.38
```

`amt`← *i finnetPresentValue vector_of_cash_flow* [Function]

Function calculates the net present value of a series of cash receipts and disbursements. The left argument is the interest rate and the right a vector of cash flow items. Conventionally, the receipts are positive and disbursement negative.

The theory is that a firm has a cost of capital, that is an average rate of both the liabilities and equity. An investment is evaluated using that rate and the expected cash flow from the investment. This calculation can be made directly from that data. Some like internal rate of return (see finirr next) are more difficult. Ussually one must take the nth root of a number and therefore one has n possible solutions. The finance workspace uses a converging iteration to find one of those solutions.

```
      .1 finnetPresentValue ¯100000 10000 11000 12000 14000 15000, 1018000
13408.07
```

`i` ← *guess finirr vector* [Function]

Internal rate of return. That is the interest rate implied by a vector of cash flows. This return is calculated iteratively using the result of the last rate of return for the current calculation. One must supply a guess to start the process.

```
      .1 finirr ¯100000 10000 11000 12000 14000 15000, 1018000
0.1222471688
```

6 fmt, a partial implementation of STSC's FMT

The fmt workspace provides a partial implementation of STSC's fmt system function. The function will format numeric data in various ways and provides an alternative to providing example formats to .

fmt takes as its left argument a character string that describes, column by column, how to render the data in an array, or number by number in a vector. This argument, we're calling the format string, is made up of one or more phrases separated by commas. Each phrase has one character that controls basic formatting. 'F' for fixed point, 'I' for integers, 'E' for scientific notation (exponential) or 'A' for characters (alpha-numeric) are examples. There are more, however they are not yet implemented in this workspace.

6.1 Structure of a phrase

Each phrase in the the format string is separated from other phrases by a comma. The phrase may contain a number for how many times the phase repeats; Modifiers with their arguments; the phrase type (E F I E A and some other letters); the width of the resulting column; and the number of decimal places or number of significant digits. Each type expects different modifiers.

For example, the phrase 3M/(/N/)/CF14.2 would be parsed as follows:

- 3 repetitions This phrase applies to the next three columns of data.
- M/(/ Negative left decorator A left parentheses will preceed each number.
- N/)/ Negative right decorator A right parentheses will follow a negative number.
- C Commas Insert comma separators between the 100s and 1,000s column; 100,000 and 1,000,000 column etc.
- F Fixed point Use fixed point formatting meaning, insert a decimal and following digits.
- 14 Width Make the width of the column 14 characters. No other white space will appear so that two numbers 101,000 and 999,650 set for a width of 9 and precision of 2 will run together.
- 2 Decimal places Each number will be displayed with two digits to the right of the decimal. Integers will show '.00'

Note the delimiters around the text argument to the M and N modifiers. The valid text delimiters will be covered in See Section 6.6 [Delimiters], page 11.

Possible modifiers vary by phrase type. See See Section 6.2 [F], page 8; See Section 6.3 [I], page 9; See Section 6.4 [E], page 10; and See Section 6.5 [A], page 10.

6.2 Fixed point

Fixed point formatting is used for numbers that may be made up of a whole number and a fractional number. The fixed point (type F) phrase must contain both the width and precision, or number of decimal places.e.g.,

```
F14.2
```

Calls for a column 14 characters wide and two places to the right of the decimal.

Fixed point phrases may also contain modifiers and should look like this:

```
rmFw.d
```

Where:

- r Repeat is the number of columns to which this phrase applies.
- m Modifiers Modifiers together with their arguments. See below
- F Phrase type Fixed Point
- w Width
- d Precision

Valid modifiers for Fixed point phrases are:

`B` *Blank if zero* [Modifier]

`C` *Comma insertion* [Modifier]

`Ki` *Scale* [Modifier]

Multiply the number by 10 raised to the ith power.

`L` *Left justify* [Modifier]

`M<text>` [Modifier]

Start each negative number with text. To differentiate positive and negative numbers one or more of M, N, P, or Q must be used.

`N<text>` [Modifier]

End each negative number with text.

`P<text>` [Modifier]

Start each positive number with text.

`Q<text>` [Modifier]

End each positive number with text.

`Z` *Zero fill* [Modifier]

The number will be padded both left and right with zeros. If M, N, P or Q is used the amount of padding will be reduced to allow room for the decorators.

6.3 Integers

Integer formatting is used for whole numbers. The decimal point will not be displayed. The fields width is required. e.g.,

```
I10
```

calls for a column ten characters wide.

Integer phrases may also contain the repetition count, and modifiers and should look like this:

```
rmIw
```

Where

- r is the number of repetition, that is columns, including the current to which the phrase applies.

- m Modifiers. As with Fixed point several modifiers are available. They are listed below
- I The integer phrase identifier.
- w Width

Valid Integer modifiers are:

`B` *Blank if zero* [Modifier]

`C` *Comma insertion* [Modifier]

`Ki` *Scale* [Modifier]
    Multiply the number by 10 raised to the ith power.

`L` *Left justify* [Modifier]

`M<text>` [Modifier]
    Start each negative number with text. To differentiate positive and negative numbers one or more the this and the following three (N P and Q) must be used.

`N<text>` [Modifier]
    End each negative number with text.

`P<text>` [Modifier]
    Start each positive number with text.

`Q<text>` [Modifier]
    End each positive number with text.

`Z` *Zero fill* [Modifier]
    The number will be padded both left and right with zeros. If M, N, P or Q is used the amount of padding will be reduced to allow room for the decorators.

6.4 Exponential

Exponential or scientific notation displays each number as a number between 0 and 10 and the exponent of 10 for the scale of the number 1500 would be

```
1.5E2
```

Both the width of the field and number of significant digits are required. A possible exponential phrase might be:

```
E10.4
```

When this phrase is applied to 1500 the result (between the vertical bars) would be

```
|   1.500E2|
```

6.5 A, Formating descriptive (text) columns

Text can be displayed with a type A phrase. How to do this depends on the data. Simple arrays require a format field for each character in a line while nested arrays are displayed in one field.

6.5.1 Simple arrays

To display a simple character array use the repeat feature:

```
          NAMES←4 6  'NUTS  SCREWSBOLTS NAILS '
          COSTS← 0.05 0.03 0.20 0.01
          QUANT← 150 200 4 1000
          '6A1,F10.2,I5,F10.2' fmt NAMES,COSTS,QUANT,[1.1]COSTS×QUANT
     NUTS        0.05  150      7.50
     SCREWS      0.03  200      6.00
     BOLTS       0.20    4      0.80
     NAILS       0.01 1000     10.00
```

Note that the number of times the type A field is repeated is equal to the width of the NAMES array.

6.5.2 Nested arrays

To display a nested character array, use the field width:

```
          NAMES← 'NUTS' 'SCREWS' 'BOLTS' 'NAILS'
          'A7,F10.2,I5,F10.2' fmt NAMES,COSTS,QUANT,[1.1]COSTS×QUANT
     NUTS         0.05  150      7.50
     SCREWS       0.03  200      6.00
     BOLTS        0.20    4      0.80
     NAILS        0.01 1000     10.00
```

6.6 Delimiting text decoration

Text that should appear within a numeric field is called a decorator. Such text should be bracketed by delimiters. There are four sets

- Decorator
- ⊂Decorato⊃
- <Decorator>
- /Decorator/

```
          '2M⊂(⊃N<)>CF14.2' fmt 13599 ¯13399
        13,599.00    (13,399.00)
```

7 html

This is a workspace to create html files. html is a text markup scheme used by the world wide web. At its most basic level html is a collection of tags, that is a word enclosed in angle brackets, which instruct a web browser how to display the text.

The html workspace is an implementation of the xml workspace. html creates a set of functions that return marked up text for inclusion in an html document. It creates a function for each html tag that takes as its optional, left argument a lexicon of attributes (see workspace lex)and as its right argument the text to be marked up.

A Hello World html document might be coded like this:

```
    head←htmlhead htmltitle 'Hello World'
    htmlhtml head  , htmlbody htmlh1 'Hello World'

<html><head><title>Hello World</title></head>
<body><h1>Hello World</h1></body></html>
```

8 import

Import is an apl workspace to import arrays from text files.

`array` ← *importfile name* [∇]

Importfile reads a file from disk and returns an array of rank 2, that is, rows and columns. It will determine whether the file is tab or comma delimited and will determine which columns contain numeric data and covert those strings to numbers.

`array` ← *importtable import_array* [∇]

Importtable examines the array returned by importfile. It will remove blank columns, heading rows at the begining of the file and footer rows at the end. It will also replace blank cells in numeric columns with zeros.

9 Lex, a name-value store

Lex is an implementation of a name-value store for apl. Functions here allow one to create such a store, add name and retrieve a value for a name.

Functions are:

`lex` ← *lexinit* [∇]
: Returns an empty store

`lex` ← *lex lexassign name value* [∇]
: Returns a new lexicon with the supplied name–value inserted into the supplied lexicon.

`item` ← *lex lexlookup name* [∇]
: Returns the value of the supplied name.

`boolean` ← *lexis lex* [∇]
: Predicate to return true if the supplied item is in fact a lexicon

`array` ← *lexkeys lex* [∇]
: Returns the list of names in the supplied lexicon

`array` ← *lexvalues lex* [∇]
: Returns a list of values in the supplied lexicon

We use and-dot-equals to do a sequential search of the list of names. For other hashing algorithms try lex1.

10 Lex1, a hashed name-value store

Lex1 is an implementation of a name-value store for apl using a hash. Functions here allow one to create such a store, add name and retrieve a value for a name.

Functions are

`lex` ← *lex1init* [∇]

Returns an empty store.

`lex` ← *lex lex1assign name–value* [∇]

Returns a hash with the supplied name–value inserted into the supplied lexicon.

`lex` ← *lex lex1lookup name* [∇]

Returns the value of the supplied name.

`boolean` ← *lex1is lex* [∇]

Predicate to return true if the supplied item is in fact a lexicon.

`array` ← *lex1keys lex* [∇]

Returns the list of names in the supplied lexicon.

`array` ← *lex1values lex* [∇]

Returns a list of values in the supplied lexicon.

`int` ← *lex1hashPrime lex key* [∇]

Returns a bucket number mapped to the supplied key.

`array` ← *lex1distribution lex* [∇]

Returns the number of items in each bucket.

We use `io+prime|+/ucs key` to compute the hash. It has two features:

- Its result is fixed and determinable for any key.
- It will yield an index into the hash.

11 cl Component files with character string indicies

This workspace supports name–value pairs in a component file. It uses the APLComponentFiles written by Blake McBride and distributed with GNU APL.

The workspaces uses the same API as lex.apl. That is the following functions:

`array` ← *clkeys lex* [∇]
: Returns a list of component names.

`file_handle` ← *clinit* [∇]
: Creates a component file ant returns a file handle.

`lex` ← *lex classign name-value* [∇]
: Assigns a component to a name. Will append or overwrite as appropriate.

`item` ← *lex cllookup name* [∇]
: Returns the component with the supplied name.

`boolean` ← *clis lex* [∇]
: Determines if the supplied file handle (an integer) is in fact a lexicon based component file.

`array` ← *clvalues lex* [∇]
: Returns all of the components of the file up to a maximum of clmax.

(`'postgresql'` *or 'sqllite'*) *clopen_db db_spec* [∇]
: Opens a database. This is a wrapper for CF_DBCONNECT. The left argument identifies the type of database while the right (db_spec) varies by that type. Postgress wants a connection string of 'host=hostname user=username password=password dbname=data_base_name' while sqlite wants a file name.

`file_handle` ← *clopen filename* [∇]
: Opens a component file and returns a file handle. clopen_db must be called before clopen as the component files are stored in an SQL database.

`clclose` *fileHandle* [∇]
: Close a component file.

`lex` ← *clclose_db* [∇]
: Closes the connection to the database. Function is a wrapper for CF_DBDISCONNECT.

12 lpr - print from APL

Workspace to print directly from APL.

`err` ← *printer lpr txt* [∇]

Function to print plain text. On success lpr returns 0, otherwise an error code greater than 0. See the man page for lpr.

Printer is a lexicon of various printing parameters. See lprUSLetter and lpra4 below.

`printAttr`←`lprUSLetter` *printer* [∇]

Function to assemble a printer lexicon for US Letter paper (8.5 inch by 11 inch). Right argument printer is the name (as CUPS understands it) of your target printer.

See your system administrator for this name.

If you are the system adiministrator and are in the dark try 'man cups'.

`printAttr` ← *lpra4 printer* [∇]

Function to Assemble a printer lexicon for A4 paper. Try '1 lprdin 'A4' for this size of A4 paper. Talk to your system administrator for the printer name.

Margins for these printer lexicons were selected to yeild printouts that I liked. If you don't, roll your own. A printer lexicon is made up of the following items:

- printer The name of the printer to use.
- pageWidth
- pageLength
- topMargin Measured in lines of text
- bottomMargin
- leftMargin Measured in characters
- rightMargin

We are printing fixed width text after all. For something fancier:

`err`← *printer lprhtml html* [∇]

Function to render and then print html. While one must supply a printer lexicon, the only item that is used here is the name of the printer.

Function returns 0 on success and something greater on failure. See the man page for lpr.

This function relies on html2ps written by Jan Kärrman; Dept. of Information Technology; Uppsala University; Sweden and is Free and Open Source Software. It is a package generally available from Ubuntu and Debian.

13 stat is short for Statistics

The stat workspace provides functions to perform statistical calculation and to organize data for statistical analysis. It is very much a work in process.

Current Functions are:

`int` ← *statcount vector* [∇]
: Returns a count of the items in the vector.

`amt` ←*slope statlmsintercept data* [∇]
: Computes the y intercept of the Least Mean Squares function given the slope the that line.

`slope` ← *statlmsslope data* [∇]
: Computes the slope of a least mean square regression. In the data (a level 2 array) data[;1] is the dependent data and data[;2] the independent.

`corr` ← *statlmscor data* [∇]
: Calculates the coefficient of correlation of the regression by use of statlmsslope and statlmsintercept.

`high_low` ← *statrange vector* [∇]
: Returns the range of a data set. The is the highest amount less the lowest.

`var` ← *statpopVar vector* [∇]
: Returns the population variance.

`sd` ← *statpopSD vector* [∇]
: Returns the population standard deviation.

`mean` ← *statmean vector* [∇]
: Returns the sample mean of a vector.

`var` ← *statsampleVar vector* [∇]
: Returns the sample variance.

`sd` ← *statsampleSD vector* [∇]
: Returns the sample standard deviation.

`median` ← *statmedian vector* [∇]
: Returns the median.

14 utl

Utl is a collection of generally usefull routines.

`utlhelpFns` *FunctionName* [∇]

Display help about a function. This routine prints the function header and any comments that immediately follow.

`utlnumberp` *item* [∇]

Tests whether item is a number. Returns true or false. See also utlnumberis.

`t←utlnumberis` *item* [∇]

Tests whether an item can become a number. IE is utlnumberp test true?

`t←utlstringp` *item* [∇]

Tests whether item is a character vector.

`new←utlstripArraySpaces` *old* [∇]

Returns a left justified array of characters with the minimum number of trailing spaces. At least one line of the array will have no trailing spaces.

`cl←utlclean` *txt* [∇]

Converts all white space to spaces and then removes duplicate spaces.

15 wp Workspace

A workspace to print arrays for an accountant. This workspace is very much a work-in-process and is included here as a test of many of the libraries published here and a test of their basic design.

Accountants have specific requirements for their work papers. They must show the company about whom the work was prepared. They must describe the work paper including the period as of which is was prepared (e.g., year ended 12/31/1957), the date of preparation, amd the author. Each of these data is stored with the underlying data as described below.

Three functions to use this system are wpinit and wpassemble and wptxtassemble. More will follow

15.1 Work Paper Lexicon

A work paper is a lexicon of many elements:

`Data`

: This is the actual array that will be printed. Make the column headings line one of the array.

`Entity`

: The name of the company, or other entity about which this work paper was prepared.

`Title`

: A General description of the work period

`Period`

: The time period of the data. Balance sheets are the balance at the end of business on a day while income statements are for a period ended on a date.

`Id` A short identifying string, like A1, B6.

`Author`

: The maker of the work paper, generally an initial.

`Attributes`

: An array the same size and shape and the data. Each cell is a lexicon which supplies the HTML attributes to guide one's browser on the display of the cell.

A special attribute `format` may be used to convert the numbers to characters. wpassemble will execute `format cell_value` in the assembly process rather than supplying the attribute to one's browser.

See wpdefaultcss below for a recipe for assembling this monster.

`Stylesheet`

: A cascading style sheet. Refer to the xml workspace documentation for how to assemble the style sheet.

There is a default style sheet, wpdefaultcss, which we recommend. It provides several classes to display various parts of your work paper. See wpdefaultcss below.

15.2 Functions in the workspace

`html` ← *wpassemble workpaper* [∇]

Returns an html page.

← *wptxtassemble wp* [txt]

Returns text. One may see the results of one's work with `←wptxtassemble workpaper`

← *wpinit 'Id'* [wp]

Create a work paper. You will be prompted for each item in the work paper lexicon. The program uses the top-quit-done paradigm:

`top` Go to the first prompt

`quit` Leave the program and abandon your work.

`done` Leave the program and return the completed work paper lexicon

`back` Go back one prompt

15.3 wpdefaultcss and its' ilk

wpdefaultcss is a cascading style sheet as implemented in the xml workspace. That is a lexicon of selectors. Each selector is itself a lexicon of css attributes that instruct the browser in how to display the select html elements. (Function xmlmkSheet returns the text document that the browser works with.)

wpdefaultcss defines a series of classes that can be assigned to a cell in one's table, viz.

```
Attr[cellrow;cellcol]←(lexinit)lexassign 'class' 'number'
```

The number class is right justified. Control the appearance of the number with format, viz.

```
Attr[cellrow;cellcol]←Attr[cellrow;cellcol] lexassign 'format' '(55,530)'
```

wpdefaultcss classes :

`colhead` Column headers. For instance:

```
Attr[1;]←⊂(lexinit)lexassign 'class' 'colhead'
```

`number` Right justified cells

`page-head`
: Special font for the heading of the work paper. That is the entity, description and period.

`initial-block`
: Special font for the author and date of the work paper

16 xml workspace

This workspace provides functions to implement an xml schema and a cascading stylesheet to display that schema.

16.1 Schema

Each element in a document using this schema can be generated from a function of that name.

Each element-function will use its left argument as the element's attributes and its right as the element's content. Attributes are stored in lexicons (see workspace lex).

For example, assume an HTML5 schema has been implemented:

```
    attr←(lexinit) lexassign 'class' 'right-justified'
    tag←attr htmlp 'Now is the winter of our discontent'
          tag
<p class="right-justified">Now is the winter of our discontent" </p>‘
```

The schema is implemented by calling xmlMkTagFns or xmlMkClosedTagFns:

16.2 Creating functions to support a schema

`xmlMkTagFns` *tag* [∇]

Creates a function for elements named tag

`tag` [xmlMkClasedTagFns]

Creates a function for a empty tag (<br\>).

We've put an example application in html_test.apl. This workspaces first provides functions for a subset of HTML5 and then defines htmlfmt_table to take an array of rank two and return an HTML page.

16.3 Cascading Stylesheets

Browsers use cascading stylesheets display an xml documnet. They consist of a lexicon (see workspace lex) of selectors and rules. Each rule consists of a lexicon of properties and values.

There are two functions for cascading stylesheets:

`lex` ← *xmlparse text* [∇]

Function returns a nested lexcion from the text of a stylesheet.

`text` ← *xmlmkSheet lex* [∇]

Function returns the text of a stylesheet from a nested lexicon.

17 The GNU Free Documentation License.

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index